UPSMON(8)

NAME

upsmon - UPS monitor and shutdown controller

SYNOPSIS

upsmon -h

upsmon -c *command*

upsmon [-D] [-K] [-p] [-u *user*]

DESCRIPTION

upsmon is the client process that is responsible for the most important part of UPS monitoring—shutting down the system when the power goes out. It can call out to other helper programs for notification purposes during power events.

upsmon can monitor multiple systems using a single process. Every UPS that is defined in the upsmon.conf(5) configuration file is assigned a power value and a type (**slave** or **master**).

OPTIONS

-h

Display the help message.

-c *command*

Send the command *command* to the existing upsmon process. Valid commands are:

fsd

shutdown all master UPSes (use with caution)

stop

stop monitoring and exit

reload

reread upsmon.conf(5) configuration file. See "reloading nuances" below if this doesn't work.

-D

Raise the debugging level. upsmon will run in the foreground and prints information on stdout about the monitoring process. Use this multiple times for more details.

-K

Test for the shutdown flag. If it exists and contains the magic string from upsmon, then upsmon will exit with `EXIT_SUCCESS`. Any other condition will make upsmon exit with `EXIT_FAILURE`.

You can test for a successful exit from `upsmon -K` in your shutdown scripts to know when to call upsdrvctl(8) to shut down the UPS.

-p

Run privileged all the time. Normally upsmon will split into two processes. The majority of the code runs as an unprivileged user, and only a tiny stub runs as root. This switch will disable that mode, and run the old "all root all the time" system.

This is not the recommended mode, and you should not use this unless you have a very good reason.

-u *user*

Set the user for the unprivileged monitoring process. This has no effect when using -p.

The default user is set at configure time with *configure --with-user=...*. Typically this is *nobody*, but other distributions will probably have a specific *nut* user for this task. If your notification scripts need to run as a specific user, set it here.

You can also set this in the upsmon.conf(5) file with the RUN_AS_USER directive.

UPS DEFINITIONS

In the upsmon.conf(5), you must specify at least one UPS that will be monitored. Use the MONITOR directive.

```
MONITOR 'system' 'powervalue' 'username' 'password' 'type'
```

The *system* refers to a upsd(8) server, in the form `upsname[@hostname[:port]]`. The default hostname is "localhost". Some examples follow:

- "su700@mybox" means a UPS called "su700" on a system called "mybox". This is the normal form.
- "fenton@bigbox:5678" is a UPS called "fenton" on a system called "bigbox" which runs upsd(8) on port "5678".

The *powervalue* refers to how many power supplies on this system are being driven this UPS. This is typically set to 1, but see the section on power values below.

The *username* is a section in your upsd.users(5) file. Whatever password you set in that section must match the *password* set in this file.

The type set in that section must also match the *type* here-- **master** or **slave**. In general, a master process is one running on the system with the UPS actually plugged into a serial port, and a slave is drawing power from the UPS but can't talk to it directly. See the section on UPS types for more.

NOTIFY EVENTS

upsmon senses several events as it monitors each UPS. They are called notify events as they can be used to tell the users and admins about the change in status. See the additional NOTIFY-related sections below for information on customizing the delivery of these messages.

ONLINE
: The UPS is back on line.

ONBATT
: The UPS is on battery.

LOWBATT
: The UPS battery is low (as determined by the driver).

FSD
: The UPS has been commanded into the "forced shutdown" mode.

COMMOK
: Communication with the UPS has been established.

COMMBAD
: Communication with the UPS was just lost.

SHUTDOWN
: The local system is being shut down.

REPLBATT
: The UPS needs to have its battery replaced.

NOCOMM
: The UPS can’t be contacted for monitoring.

NOTIFY COMMAND

In upsmon.conf(5), you can configure a program called the NOTIFYCMD that will handle events that occur.

`NOTIFYCMD` "*path to program*"

```
NOTIFYCMD "/usr/local/bin/notifyme"
```

Remember to wrap the path in "quotes" if it contains any spaces.

The program you run as your NOTIFYCMD can use the environment variables NOTIFYTYPE and UPSNAME to know what has happened and on which UPS. It also receives the notification message (see below) as the first (and only) argument, so you can deliver a preformatted message too.

Note that the NOTIFYCMD will only be called for a given event when you set the EXEC flag by using the notify flags, below:

NOTIFY FLAGS

By default, all notify events (see above) generate a global message (wall) to all users, plus they are logged via the syslog. You can change this with the NOTIFYFLAG directive in the configuration file:

`NOTIFYFLAG` *notifytype flags*

Examples:

- `NOTIFYFLAG ONLINE SYSLOG`
- `NOTIFYFLAG ONBATT SYSLOG+WALL`
- `NOTIFYFLAG LOWBATT SYSLOG+WALL+EXEC`

The flags that can be set on a given notify event are:

SYSLOG
  Write this message to the syslog.

WALL
  Send this message to all users on the system via **wall**(1).

EXEC
  Execute the NOTIFYCMD.

IGNORE
  Don't do anything. If you use this, don't use any of the other flags.

You can mix these flags. "SYSLOG+WALL+EXEC" does all three for a given event.

NOTIFY MESSAGES

upsmon comes with default messages for each of the NOTIFY events. These can be changed with the NOTIFYMSG directive.

`NOTIFYMSG` *type* "*message*"

Examples:

- `NOTIFYMSG ONLINE "UPS %s is getting line power"`
- ` NOTIFYMSG ONBATT "Someone pulled the plug on %s"`

The first instance of %s is replaced with the identifier of the UPS that generated the event. These messages are used when sending walls to the users directly from upsmon, and are also passed to the NOTIFYCMD.

POWER VALUES

The "current overall power value" is the sum of all UPSes that are currently able to supply power to the system hosting upsmon. Any UPS that is either on line or just on battery contributes to this number. If a UPS is critical (on battery and low battery) or has been put into "forced shutdown" mode, it no longer contributes.

A "power value" on a MONITOR line in the config file is the number of power supplies that the UPS runs on the current system.

MONITOR *upsname powervalue username password type*

Normally, you only have one power supply, so it will be set to 1.

```
MONITOR myups@myhost 1 username mypassword master
```

On a large server with redundant power supplies, the power value for a UPS may be greater than 1. You may also have more than one of them defined.

```
MONITOR ups-alpha@myhost 2 username mypassword master
```

```
MONITOR ups-beta@myhost 2 username mypassword master
```

You can also set the power value for a UPS to 0 if it does not supply any power to that system. This is generally used when you want to use the upsmon notification features for a UPS even though it's not actually running the system that hosts upsmon. Don't set this to "master" unless you really want to power this UPS off when this instance of upsmon needs to shut down for its own reasons.

```
MONITOR faraway@anotherbox 0 username mypassword slave
```

The "minimum power value" is the number of power supplies that must be receiving power in order to keep the computer running.

MINSUPPLIES *value*

Typical PCs only have 1, so most users will leave this at the default.

```
MINSUPPLIES 1
```

If you have a server or similar system with redundant power, then this value will usually be set higher. One that requires three power supplies to be running at all times would simply set it to 3.

```
MINSUPPLIES 3
```

When the current overall power value drops below the minimum power value, upsmon starts the shutdown sequence. This design allows you to lose some of your power supplies in a redundant power environment without bringing down the entire system while still working properly for smaller systems.

UPS TYPES

upsmon and upsd(8) don't always run on the same system. When they do, any UPSes that are directly attached to the upsmon host should be monitored in "master" mode. This makes upsmon take charge of that equipment, and it will wait for slaves to disconnect before shutting down the local system. This allows the distant systems (monitoring over the network) to shut down cleanly before `upsdrvctl shutdown` runs and turns them all off.

When upsmon runs as a slave, it is relying on the distant system to tell it about the state of the UPS. When that UPS goes critical (on battery and low battery), it immediately invokes the local shutdown command. This needs to happen quickly. Once it disconnects from the distant upsd(8) server, the master upsmon will start its own shutdown process. Your slaves must all shut down before the master turns off the power or filesystem damage may result.

upsmon deals with slaves that get wedged, hang, or otherwise fail to disconnect from upsd(8) in a timely manner with the HOSTSYNC timer. During a shutdown situation, the master upsmon will give up after this interval and it will shut down anyway. This keeps the master from sitting there forever (which would endanger that host) if a slave should break somehow. This defaults to 15 seconds.

If your master system is shutting down too quickly, set the FINALDELAY interval to something greater than the default 15 seconds. Don't set this too high, or your UPS battery may run out of power before the master upsmon process shuts down that system.

TIMED SHUTDOWNS

For those rare situations where the shutdown process can't be completed between the time that low battery is signalled and the UPS actually powers off the load, use the upssched(8) helper program. You can use it along with upsmon to schedule a shutdown based on the "on battery" event. upssched can then come back to upsmon to initiate the shutdown once it has run on battery too long.

This can be complicated and messy, so stick to the default critical UPS handling if you can.

REDUNDANT POWER SUPPLIES

If you have more than one power supply for redundant power, you may also have more than one UPS feeding your computer. upsmon can handle this. Be sure to set the UPS power values appropriately and the MINSUPPLIES value high enough so that it keeps running until it really does need to shut down.

For example, the HP NetServer LH4 by default has 3 power supplies installed, with one bay empty. It has two power cords, one per side of the box. This means that one power cord powers two power supply bays, and that you can only have two UPSes supplying power.

Connect UPS "alpha" to the cord feeding two power supplies, and UPS "beta" to the cord that feeds the third and the empty slot. Define alpha as a powervalue of 2, and beta as a powervalue of 1. Set the MINSUPPLIES to 2.

When alpha goes on battery, your current overall power value will stay at 3, as it's still supplying power. However, once it goes critical (on battery and low battery), it will stop contributing to the current overall power value. That means the value will be 1 (beta alone), which is less than 2. That is insufficient to run the system, and upsmon will invoke the shutdown sequence.

However, if beta goes critical, subtracting its contribution will take the current overall value from 3 to 2. This is just high enough to satisfy the minimum, so the system will continue running as before. If beta returns later, it will be re-added and the current value will go back to 3. This allows you to swap out UPSes, change a power configuration, or whatever, as long as you maintain the minimum power value at all times.

MIXED OPERATIONS

Besides being able to monitor multiple UPSes, upsmon can also monitor them as different roles. If you have a system with multiple power supplies serviced by separate UPS batteries, it’s possible to be a master on one and a slave on the other. This usually happens when you run out of serial ports and need to do the monitoring through another system nearby.

This is also complicated, especially when it comes time to power down a UPS that has gone critical but doesn’t supply the local system. You can do this with some scripting magic in your notify command script, but it’s beyond the scope of this manual.

FORCED SHUTDOWNS

When upsmon is forced to bring down the local system, it sets the "FSD" (forced shutdown) flag on any UPSes that it is running in master mode. This is used to synchronize slaves in the event that a master UPS that is otherwise OK needs to be brought down due to some pressing event on the master.

You can manually invoke this mode on the master upsmon by starting another copy with `-c fsd`. This is useful when you want to initiate a shutdown before the critical stage through some external means, such as upssched(8).

DEAD UPSES

In the event that upsmon can’t reach upsd(8), it declares that UPS "dead" after some interval controlled by DEADTIME in the upsmon.conf(5). If this happens while that UPS was last known to be on battery, it is assumed to have gone critical and no longer contributes to the overall power value.

upsmon will alert you to a UPS that can’t be contacted for monitoring with a "NOCOMM" notifier by default every 300 seconds. This can be changed with the NOCOMMWARNTIME setting.

RELOADING NUANCES

upsmon usually gives up root powers for the process that does most of the work, including handling signals like SIGHUP to reload the configuration file. This means your upsmon.conf(8) file must be readable by the non-root account that upsmon switches to.

If you want reloads to work, upsmon must run as some user that has permissions to read the configuration file. I recommend making a new user just for this purpose, as making the file readable by "nobody" (the default user) would be a bad idea.

See the RUN_AS_USER section in upsmon.conf(8) for more on this topic.

Additionally, you can’t change the SHUTDOWNCMD or POWERDOWNFLAG definitions with a reload due to the split-process model. If you change those values, you **must** stop upsmon and start it back up. upsmon will warn you in the syslog if you make changes to either of those values during a reload.

SIMULATING POWER FAILURES

To test a synchronized shutdown without pulling the plug on your UPS(es), you need only set the forced shutdown (FSD) flag on them. You can do this by calling upsmon again to set the flag, i.e.:

```
upsmon -c fsd
```

After that, the master and the slaves will do their usual shutdown sequence as if the battery had gone critical. This is much easier on your UPS equipment, and it beats crawling under a desk to find the plug.

FILES

upsmon.conf(5)

SEE ALSO

Server:

upsd(8)

Clients:

upsc(8), upscmd(8), upsrw(8), upsmon(8)

CGI programs:

upsset.cgi(8), upsstats.cgi(8), upsimage.cgi(8)

Internet resources:

The NUT (Network UPS Tools) home page: http://www.networkupstools.org/

Last updated 2016-03-09 14:05:25 CET -- Network UPS Tools 2.7.4